

# **Exhibit T-4**

U.S. Patent

Feb. 9, 2010

Sheet 211 of 403

US 7,660,973 B2

**Definition**

```

def AddressImmediate(op,rd,rc,imm) as
  i ← imm17 || imm
  c ← RegRead(rc, 64)
  case op of
    A.SUB.I:
      a ← i - c
    A.SUB.I.O:
      t ← (i63 || i) - (c63 || c)
      if t64 ≠ t63 then
        raise FixedPointArithmetic
      endif
      a ← t63..0
    A.SUB.I.U.O:
      t ← (i63 || i) - (c63 || c)
      if t64 ≠ 0 then
        raise FixedPointArithmetic
      endif
      a ← t63..0
    A.SET.AND.E.I:
      a ← ((i and c) = 0)64
    A.SET.AND.NE.I:
      a ← ((i and c) ≠ 0)64
    A.SET.E.I:
      a ← (i = c)64
    A.SET.NE.I:
      a ← (i ≠ c)64
    A.SET.L.I:
      a ← (i < c)64
    A.SET.GE.I:
      a ← (i ≥ c)64
    A.SET.L.I.U:
      a ← ((0 || i) < (0 || c))64
    A.SET.GE.I.U:
      a ← ((0 || i) ≥ (0 || c))64
  endcase
  RegWrite(rd, 64, a)
enddef

```

**Exceptions**

Fixed-point arithmetic

FIG. 63C

U.S. Patent

Feb. 9, 2010

Sheet 212 of 403

US 7,660,973 B2

## Operation codes

A.SET.AND.E	Address set and equal zero
A.SET.AND.NE	Address set and not equal zero
A.SET.E	Address set equal
A.SET.GE	Address set greater equal signed
A.SET.GE.U	Address set greater equal unsigned
A.SET.L	Address set less signed
A.SET.L.U	Address set less unsigned
A.SET.NE	Address set not equal
A.SUB	Address subtract
A.SUB.O	Address subtract signed check overflow
A.SUB.U.O	Address subtract unsigned check overflow

## Equivalencies

A.SET.E.Z	Address set equal zero
A.SET.G.Z	Address set greater zero signed
A.SET.GE.Z	Address set greater equal zero signed
A.SET.L.Z	Address set less zero signed
A.SET.LE.Z	Address set less equal zero signed
A.SET.NE.Z	Address set not equal zero
A.SET.G	Address set greater signed
A.SET.G.U	Address set greater unsigned
A.SET.LE	Address set less equal signed
A.SET.LE.U	Address set less equal unsigned

A.SET.E.Z rd=rc	← A.SET.AND.E rd=rc,rc
A.SET.G.Z rd=rc	← A.SET.L.U rd=rc,rc
A.SET.GE.Z rd=rc	← A.SET.GE rd=rc,rc
A.SET.L.Z rd=rc	← A.SET.L rd=rc,rc
A.SET.LE.Z rd=rc	← A.SET.GE.U rd=rc,rc
A.SET.NE.Z rd=rc	← A.SET.AND.NE rd=rc,rc
A.SET.G rd=rb,rc	→ A.SET.L rd=rc,rb
A.SET.G.U rd=rb,rc	→ A.SET.L.U rd=rc,rb
A.SET.LE rd=rb,rc	→ A.SET.GE rd=rc,rb
A.SET.LE.U rd=rb,rc	→ A.SET.GE.U rd=rc,rb

## Redundancies

A.SET.E rd=rc,rc	⇔ A.SET rd
A.SET.NE rd=rc,rc	⇔ A.ZERO rd

FIG. 64A

U.S. Patent

Feb. 9, 2010

Sheet 213 of 403

US 7,660,973 B2

## Selection

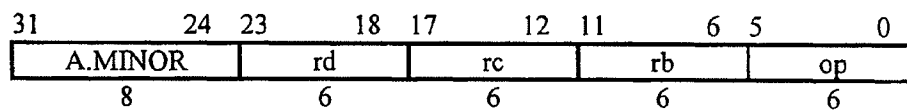
class	operation	cond	operand	check
arithmetic	SUB			
			NONE U	O
boolean	SET.AND SET	E NE		
	SET	L GE G LE	NONE U	
	SET	L GE G LE E NE	Z	

## Format

op rd=rb,rc

rd=op(rb,rc)

rd=opz(rcb)



rc ← rb ← rcb

FIG. 64B

U.S. Patent

Feb. 9, 2010

Sheet 214 of 403

US 7,660,973 B2

**Definition**

```

def AddressReversed(op,rd,rc,rb) as
  c ← RegRead(rc, 128)
  b ← RegRead(rb, 128)
  case op of
    A.SET.E:
      a ← (b = c)64
    A.SET.NE:
      a ← (b ≠ c)64
    A.SET.AND.E:
      a ← ((b and c) = 0)64
    A.SET.AND.NE:
      a ← ((b and c) ≠ 0)64
    A.SET.L:
      a ← ((rc = rb) ? (b < 0) : (b < c))64
    A.SET.GE:
      a ← ((rc = rb) ? (b ≥ 0) : (b ≥ c))64
    A.SET.L.U:
      a ← ((rc = rb) ? (b > 0) : ((0 || b) < (0 || c)))64
    A.SET.GE.U:
      a ← ((rc = rb) ? (b ≤ 0) : ((0 || b) ≥ (0 || c)))64
    A.SUB:
      a ← b - c
    A.SUB.O:
      t ← (b63 || b) - (c63 || c)
      if t64 ≠ t63 then
        raise FixedPointArithmetic
      endif
      a ← t63..0
    A.SUB.U.O:
      t ← (01 || b) - (01 || c)
      if t64 ≠ 0 then
        raise FixedPointArithmetic
      endif
      a ← t63..0
  endcase
  RegWrite(rd, 64, a)
enddef

```

**Exceptions**

Fixed-point arithmetic

FIG. 64C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 215 of 403**

**US 7,660,973 B2**

**Operation codes**

A.SHL.I.ADD	Address shift left immediate add
-------------	----------------------------------

FIG. 65A

**U.S. Patent**

Feb. 9, 2010

Sheet 216 of 403

**US 7,660,973 B2****Format**

A.SHL.I.ADD rd=rc,rb,i

rc=op(ra,rb,i)

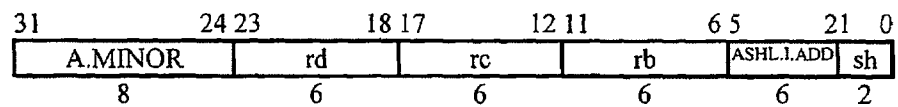
assert  $1 \leq i \leq 4$ sh  $\leftarrow$  i-1

FIG. 65B

**U.S. Patent**

Feb. 9, 2010

Sheet 217 of 403

**US 7,660,973 B2**

**Definition**

```
def AddressShiftLeftImmediateAdd(sh,rd,rc,rb) as
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 64)
  a ← c + (b62-sh..0 || 01+sh)
  RegWrite(rd, 64, a)
enddef
```

**Exceptions**

none

FIG. 65C



**U.S. Patent**

**Feb. 9, 2010**

**Sheet 218 of 403**

**US 7,660,973 B2**

**Operation codes**

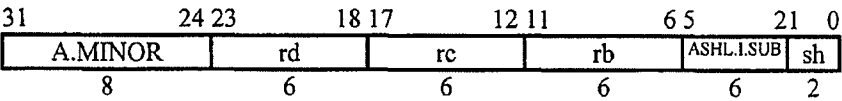
A.SHL.I.SUB	Address shift left immediate subtract
-------------	---------------------------------------

**FIG. 66A**

Format

ASHL.I.SUB rd=rb,i,rc

rd=op(rb,i,rc)



assert  $1 \leq i \leq 4$

sh  $\leftarrow$  i-1

FIG. 66B

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 220 of 403**

**US 7,660,973 B2**

**Definition**

```
def AddressShiftLeftImmediateSubtract(op,rd,rc,rb) as
  c ← RegRead(rc, 128)
  b ← RegRead(rb, 128)
  a ← (b62-sh..0 || 01+sh) - c
  RegWrite(rd, 64, a)
enddef
```

**Exceptions**

none

**FIG. 66C**

**U.S. Patent**

Feb. 9, 2010

Sheet 221 of 403

**US 7,660,973 B2****Operation codes**

A.SHL.I	Address shift left immediate
A.SHL.I.O	Address shift left imMediate signed check overflow
A.SHL.I.U.O	Address shift left immediate unsigned check overflow
A.SHR.I	Address signed shift right immediate
A.SHR.I.U	Address shift right immediate unsigned

**Redundancies**

A.SHL.I rd=rc,1	⇔	A.ADD rd=rc,rc
A.SHL.I.O rd=rc,1	⇔	A.ADD.O rd=rc,rc
A.SHL.I.U.O rd=rc,1	⇔	A.ADD.U.O rd=rc,rc
A.SHL.I rd=rc,0	⇔	A.COPY rd=rc
A.SHL.I.O rd=rc,0	⇔	A.COPY rd=rc
A.SHL.I.U.O rd=rc,0	⇔	A.COPY rd=rc
A.SHR.I rd=rc,0	⇔	A.COPY rd=rc
A.SHR.I.U rd=rc,0	⇔	A.COPY rd=rc

FIG. 67A

Selection

class	operation	form	operand	check
shift	SHL	I		
			NONE U	O
	SHR	I	NONE U	

Format

op    rd=rc,simm

rd=op(rc,simm)

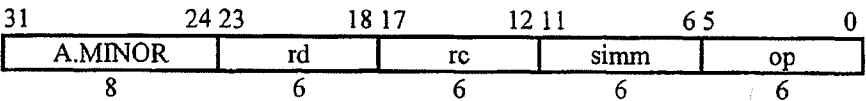


FIG. 67B

U.S. Patent

Feb. 9, 2010

Sheet 223 of 403

US 7,660,973 B2

**Definition**

```

def AddressShiftImmediate(op,rd,rc,simm) as
  c ← RegRead(rc, 64)
  case op of
    A.SHL.I:
      a ← c63-simm..0 || 0simm
    A.SHL.I.O:
      if c63..63-simm ≠ c63simm+1 then
        raise FixedPointArithmetic
      endif
      a ← c63-simm..0 || 0simm
    A.SHL.I.U.O:
      if c63..64-simm ≠ 0 then
        raise FixedPointArithmetic
      endif
      a ← c63-simm..0 || 0simm
    A.SHR.I:
      a ← a63simm || c63..simm
    A.SHR.I.U:
      a ← 0simm || c63..simm
  endcase
  RegWrite(rd, 64, a)
enddef

```

**Exceptions**

Fixed-point arithmetic

FIG. 67C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 224 of 403**

**US 7,660,973 B2**

**Operation codes**

A.MUX	Address multiplex
-------	-------------------

FIG. 68A

Format

op    ra=rd,rc,rb

ra=amux(rd,rc,rb)

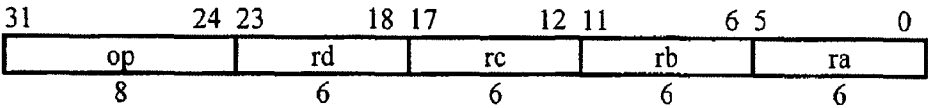


FIG. 68B



**U.S. Patent**

Feb. 9, 2010

Sheet 226 of 403

**US 7,660,973 B2**

### **Definition**

```
def AddressTernary(op,rd,rc,rb,ra) as
  d ← RegRead(rd, 64)
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 64)
  endcase
  case op of
    A.MUX:
      a ← (c and d) or (b and not d)
  endcase
  RegWrite(ra, 64, a)
enddef
```

### **Exceptions**

none

**FIG. 68C**

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 227 of 403**

**US 7,660,973 B2**

**Operation codes**

<b>B</b>	<b>Branch</b>
----------	---------------

**FIG. 69A**

Format

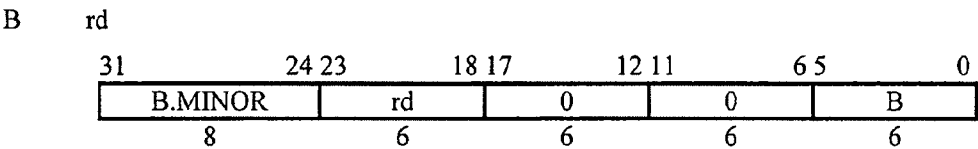


FIG. 69B

**U.S. Patent**

Feb. 9, 2010

Sheet 229 of 403

**US 7,660,973 B2****Definition**

```
def Branch(rd,rc,rb) as
  if (rc ≠ 0) or (rb ≠ 0) then
    raise ReservedInstruction
  endif
  d ← RegRead(rd, 64)
  if (d1..0) ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  ProgramCounter ← d63..2 || 02
  raise TakenBranch
enddef
```

**Exceptions**

Reserved Instruction

Access disallowed by virtual address

FIG. 69C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 230 of 403**

**US 7,660,973 B2**

**Operation codes**

B.BACK	Branch back
--------	-------------

**FIG. 70A**

Format

B.BACK

bback()

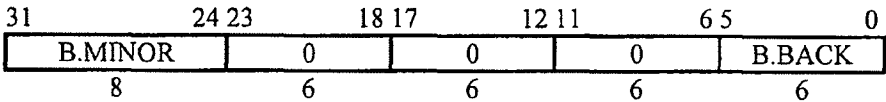


FIG. 70B

**U.S. Patent**

Feb. 9, 2010

Sheet 232 of 403

**US 7,660,973 B2****Definition**

```

def BranchBack(rd,rc,rb) as
  c ← RegRead(rc, 128)
  if (rd ≠ 0) or (rc ≠ 0) or (rb ≠ 0) then
    raise ReservedInstruction
  endif
  a ← LoadMemory(ExceptionBase,ExceptionBase+Thread*128,128,L)
  if PrivilegeLevel > c1..0 then
    PrivilegeLevel ← c1..0
  endif
  ProgramCounter ← c63..2 || 02
  ExceptionState ← 0
  RegWrite(rd,128,a)
  raise TakenBranchContinue
enddef

```

**Exceptions**

Reserved Instruction  
 Access disallowed by virtual address  
 Access disallowed by tag  
 Access disallowed by global TB  
 Access disallowed by local TB  
 Access detail required by tag  
 Access detail required by local TB  
 Access detail required by global TB  
 Local TB miss  
 Global TB miss

FIG. 70C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 233 of 403**

**US 7,660,973 B2**

**Operation codes**

B.BARRIER	Branch barrier
-----------	----------------

**FIG. 71A**



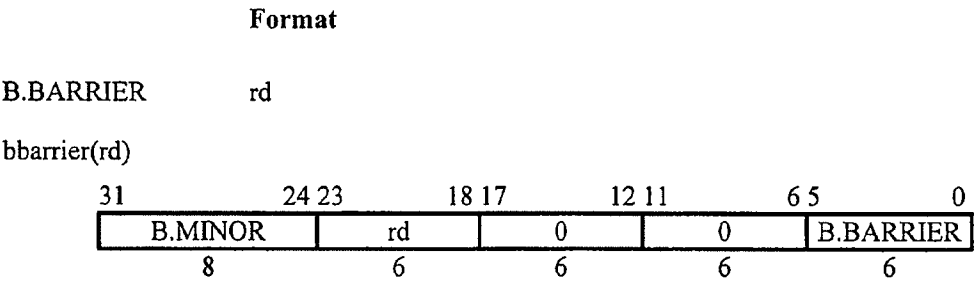


FIG. 71B

**U.S. Patent**

Feb. 9, 2010

Sheet 235 of 403

**US 7,660,973 B2****Definition**

```

def BranchBarrier(rd,rc,rb) as
  if (rc ≠ 0) or (rb ≠ 0) then
    raise ReservedInstruction
  endif
  d ← RegRead(rd, 64)
  if (d1..0) ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  ProgramCounter ← d63..2 || 02
  FetchBarrier()
  raise TakenBranch
enddef

```

**Exceptions**

Reserved Instruction

FIG. 71C

U.S. Patent

Feb. 9, 2010

Sheet 236 of 403

US 7,660,973 B2

## Operation codes

B.AND.E	Branch and equal zero
B.AND.NE	Branch and not equal zero
B.E	Branch equal
B.GE	Branch greater equal signed
B.L	Branch signed less
B.NE	Branch not equal
B.GE.U	Branch greater equal unsigned
B.L.U	Branch less unsigned

## Equivalencies

<i>B.E.Z</i>	Branch equal zero
<i>B.G.Z<sup>1</sup></i>	Branch greater zero signed
<i>B.GE.Z<sup>2</sup></i>	Branch greater equal zero signed
<i>B.L.Z<sup>3</sup></i>	Branch less zero signed
<i>B.LE.Z<sup>4</sup></i>	Branch less equal zero signed
<i>B.NE.Z</i>	Branch not equal zero
<i>B.LE</i>	Branch less equal signed
<i>B.G</i>	Branch greater signed
<i>B.LE.U</i>	Branch less equal unsigned
<i>B.G.U</i>	Branch greater unsigned
<i>B.NOP</i>	Branch no operation

<i>B.E.Z rc,target</i>	←	B.AND.E rc,rc,target
<i>B.G.Z rc,target</i>	←	B.L.U rc,rc,target
<i>B.GE.Z rc,target</i>	←	B.GE rc,rc,target
<i>B.L.Z rc,target</i>	←	B.L rc,rc,target
<i>B.LE.Z rc,target</i>	←	B.GE.U rc,rc,target
<i>B.NE.Z rc,target</i>	←	B.AND.NE rc,rc,target
<i>B.LE rc,rd,target</i>	→	B.GE rd,rc,target
<i>B.G rc,rd,target</i>	→	B.L rd,rc,target
<i>B.LE.U rc,rd,target</i>	→	B.GE.U rd,rc,target
<i>B.G.U rc,rd,target</i>	→	B.L.U rd,rc,target
<i>B.NOP</i>	←	B.NE r0,r0,\$

## Redundancies

B.E rc,rc,target	↔	B.I target
B.NE rc,rc,target	↔	B.NOP

FIG. 72A

## Selection

class	op	compare				type	
arithmetic		L	GE	G	LE	NONE	U
vs. zero		L	GE	G	LE	Z	
			E	NE			
bitwise	none AND	E	NE				

### Format

op      rd,rc,target

```
if (op(rd,rc)) goto target;
```

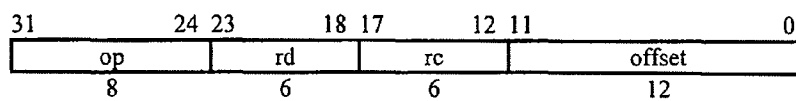


FIG. 72B

**U.S. Patent**

Feb. 9, 2010

Sheet 238 of 403

**US 7,660,973 B2****Definition**

```

def BranchConditionally(op,rd,rc,offset) as
  d ← RegRead(rd, 128)
  c ← RegRead(rc, 128)
  case op of
    B.E:
      a ← d = c
    B.NE:
      a ← d ≠ c
    B.AND.E:
      a ← (d and c) = 0
    BAND.NE:
      a ← (d and c) ≠ 0
    B.L:
      a ← (rd = rc) ? (c < 0): (d < c)
    B.GE:
      a ← (rd = rc) ? (c ≥ 0): (d ≥ c)
    B.L.U:
      a ← (rd = rc) ? (c > 0): ((0 ∥ d) < (0 ∥ c))
    B.GE.U:
      a ← (rd = rc) ? (c ≤ 0): ((0 ∥ d) ≥ (0 ∥ c))
  endcase
  if a then
    ProgramCounter ← ProgramCounter + (offset50 ∥ offset ∥ 02)
    raise TakenBranch
  endif
enddef

```

**Exceptions**

none

FIG. 72C

U.S. Patent

Feb. 9, 2010

Sheet 239 of 403

US 7,660,973 B2

## Operation codes

B.E.F. 16	Branch equal floating-point half
B.E.F. 32	Branch equal floating-point single
B.E.F. 64	Branch equal floating-point double
B.E.F.128	Branch equal floating-point quad
B.GE.F. 16	Branch greater equal floating-point half
B.GE.F. 32	Branch greater equal floating-point single
B.GE.F. 64	Branch greater equal floating-point double
B.GE.F.128	Branch greater equal floating-point quad
B.L.F. 16	Branch less floating-point half
B.L.F. 32	Branch less floating-point single
B.L.F. 64	Branch less floating-point double
B.L.F.128	Branch less floating-point quad
B.LG.F. 16	Branch less greater floating-point half
B.LG.F. 32	Branch less greater floating-point single
B.LG.F. 64	Branch less greater floating-point double
B.LG.F.128	Branch less greater floating-point quad

## Equivalencies

<i>B.LE.F. 16</i>	Branch less equal floating-point half
<i>B.LE.F. 32</i>	Branch less equal floating-point single
<i>B.LE.F. 64</i>	Branch less equal floating-point double
<i>B.LE.F.128</i>	Branch less equal floating-point quad
<i>B.G.F. 16</i>	Branch greater floating-point half
<i>B.G.F. 32</i>	Branch greater floating-point single
<i>B.G.F. 64</i>	Branch greater floating-point double
<i>B.G.F.128</i>	Branch greater floating-point quad

<i>B.LE.F.size rc,rd,target</i>	→ B.GE.F.size rd,rc,target
<i>B.G.F.size rc,rd,target</i>	→ B.L.F.size rd,rc,target

FIG. 73A

FIG. 73B

U.S. Patent

Feb. 9, 2010

Sheet 241 of 403

US 7,660,973 B2

**Definition**

```

def BranchConditional(FloatingPointop,rd,rc,offset) as
  case op of
    B.E.F.16, B.LG.F.16, B.L.F.16, B.GE.F.16:
      size ← 16
    B.E.F.32, B.LG.F.32, B.L.F.32, B.GE.F.32:
      size ← 32
    B.E.F.64, B.LG.F.64, B.L.F.64, B.GE.F.64:
      size ← 64
    B.E.F.128, B.LG.F.128, B.L.F.128, B.GE.F.128:
      size ← 128
  endcase
  d ← F(size,RegRead(rd, 128))
  c ← F(size,RegRead(rc, 128))
  v ← fcom(d, c)
  case op of
    BEF16, BEF32, BEF64, BEF128:
      a ← (v = E)
    BLGF16, BLGF32, BLGF64, BLGF128:
      a ← (v = L) or (v = G)
    BLF16, BLF32, BLF64, BLF128:
      a ← (v = L)
    BGEF16, BGEF32, BGEF64, BGEF128:
      a ← (v = G) or (v = E)
  endcase
  if a then
    ProgramCounter ← ProgramCounter + (offset50 || offset || 02)
    raise TakenBranch
  endif
enddef

```

**Exceptions**

none

FIG. 73C



**U.S. Patent**

**Feb. 9, 2010**

**Sheet 242 of 403**

**US 7,660,973 B2**

**Operation codes**

B.I.F. 32	Branch invisible floating-point single
B.NI.F. 32	Branch not invisible floating-point single
B.NV.F. 32	Branch not visible floating-point single
B.V.F. 32	Branch visible floating-point single

**FIG. 74A**

Selection

number format	type	compare	size
floating-point	F	I    NI    NV    V	32

Format

op    rc,rd,target

if (op(rc,rd)) goto target;

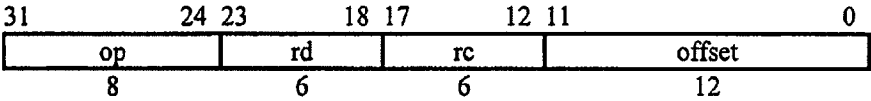


FIG. 74B

**U.S. Patent**

Feb. 9, 2010

Sheet 244 of 403

**US 7,660,973 B2****Definition**

```
def n(a) as (a.t=QNAN) or (a.t=SNAN) enddef
```

```
def less(a,b) as fcom(a,b)=L enddef
```

```
def trxya,b,c,d) as (fcom(fabs(a),b)=G) and (fcom(fabs(c),d)=G) and (a.s=c.s) enddef
```

```
def BranchConditionalVisibilityFloatingPoint(op,rd,rc,offset) as
```

```
  d ← RegRead(rd, 128)
```

```
  c ← RegRead(rc, 128)
```

```
  dx ← F(32,d31..0)
```

```
  cx ← F(32,c31..0)
```

```
  dy ← F(32,d63..32)
```

```
  cy ← F(32,c63..32)
```

```
  dz ← F(32,d95..64)
```

```
  cz ← F(32,c95..64)
```

```
  dw ← F(32,d127..96)
```

```
  cw ← F(32,c127..96)
```

```
  f1 ← F(32,0x7f000000) // floating-point 1.0
```

```
  if (n(dx) or n(dy) or n(dz) or n(dw) or n(cx) or n(cy) or n(cz) or n(cw)) then
```

```
    a ← false
```

```
  else
```

```
    dv ← less(fabs(dx),dz) and less(fabs(dy),dz) and less(dz,f1) and (dz.s=0)
```

```
    cv ← less(fabs(cx),cz) and less(fabs(cy),cz) and less(cz,f1) and (cz.s=0)
```

```
    trz ← (less(f1,dz) and less(f1,cz)) or ((dz.s=1 and cz.s=1))
```

```
    tr ← trxy(dx,dz,cx,cz) or trxy(dy,dz,cy,cz) or trz
```

```
    case op of
```

```
      B.I.F.32:
```

```
        a ← tr
```

```
      B.NI.F.32:
```

```
        a ← not tr
```

```
      B.NV.F.32:
```

```
        a ← not (dv and cv)
```

```
      B.V.F.32:
```

```
        a ← dv and cv
```

```
    endcase
```

```
  endif
```

```
  if a then
```

```
    ProgramCounter ← ProgramCounter + (offset50 || offset || 02)
```

FIG. 74C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 245 of 403**

**US 7,660,973 B2**

```
        raise TakenBranch
    endif
enddef

    Exceptions

none
```

**FIG. 74C *continued***

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 246 of 403**

**US 7,660,973 B2**

**Operation codes**

<b>B.DOWN</b>	<b>Branch down</b>
---------------	--------------------

**FIG. 75A**

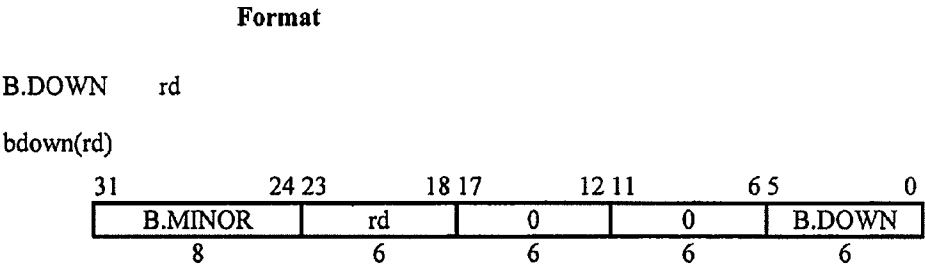


FIG. 75B

**U.S. Patent**

Feb. 9, 2010

Sheet 248 of 403

**US 7,660,973 B2****Definition**

```
def BranchDown(rd,rc,rb) as
  if (rc  $\neq$  0) or (rb  $\neq$  0) then
    raise ReservedInstruction
  endif
  d  $\leftarrow$  RegRead(rd, 64)
  if PrivilegeLevel > d1..0 then
    PrivilegeLevel  $\leftarrow$  d1..0
  endif
  ProgramCounter  $\leftarrow$  d63..2 || 02
  raise TakenBranch
enddef
```

**Exceptions****Reserved Instruction****FIG. 75C**

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 249 of 403**

**US 7,660,973 B2**

**Operation codes**

B.GATE	Branch gateway
--------	----------------

**Equivalencies**

B.GATE	← B.GATE 0
--------	------------

**FIG. 76A**



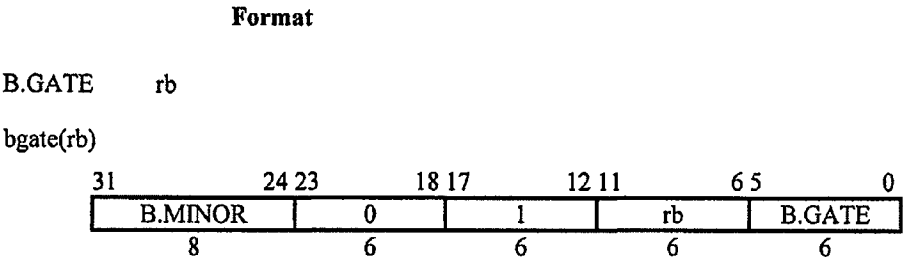


FIG. 76B

**U.S. Patent**

Feb. 9, 2010

Sheet 251 of 403

**US 7,660,973 B2****Definition**

```

def BranchGateway(rd,rc,rb) as
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 64)
  if (rd ≠ 0) or (rc ≠ 1) then
    raise ReservedInstruction
  endif
  if c2..0 ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  d ← ProgramCounter63..2+1 || PrivilegeLevel
  if PrivilegeLevel < b1..0 then
    m ← LoadMemoryG(c,c,64,L)
    if b ≠ m then
      raise GatewayDisallowed
    endif
    PrivilegeLevel ← b1..0
  endif
  ProgramCounter ← b63..2 || 02
  RegWrite(rd, 64, d)
  raise TakenBranch
enddef

```

**Exceptions**

Reserved Instruction  
 Gateway disallowed  
 Access disallowed by virtual address  
 Access disallowed by tag  
 Access disallowed by global TB  
 Access disallowed by local TB  
 Access detail required by tag  
 Access detail required by local TB  
 Access detail required by global TB  
 Local TB miss  
 Global TB miss

FIG. 76C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 252 of 403**

**US 7,660,973 B2**

**Operation codes**

B.HALT	Branch halt
--------	-------------

**FIG. 77A**

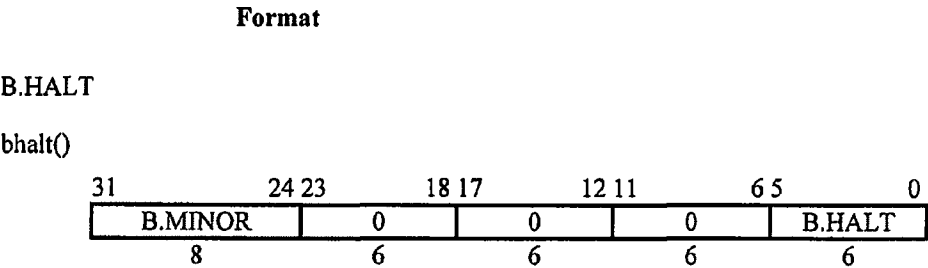


FIG. 77B

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 254 of 403**

**US 7,660,973 B2**

**Definition**

```
def BranchHalt(rd,rc,rb) as
    if (rd  $\neq$  0) or (rc  $\neq$  0) or (rb  $\neq$  0) then
        raise ReservedInstruction
    endif
    FetchHalt()
enddef
```

**Exceptions**

Reserved Instruction

**FIG. 77C**

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 255 of 403**

**US 7,660,973 B2**

**Operation codes**

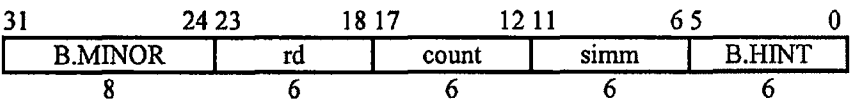
B.HINT	Branch Hint
--------	-------------

**FIG. 78A**

Format

B.HINT      badd,count,rd

bhint(badd,count,rd)



simm ← badd-pc-4

FIG. 78B

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 257 of 403**

**US 7,660,973 B2**

**Definition**

```
def BranchHint(rd,count,simm) as
  d ← RegRead(rd, 64)
  if (d1..0) ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  FetchHint(ProgramCounter +4 + (0 || simm || 02), d63..2 || 02, count)
enddef
```

**Exceptions**

Access disallowed by virtual address

**FIG. 78C**



**U.S. Patent**

**Feb. 9, 2010**

**Sheet 258 of 403**

**US 7,660,973 B2**

**Operation codes**

<b>B.HINT.I</b>	<b>Branch Hint Immediate</b>
-----------------	------------------------------

**FIG. 79A**

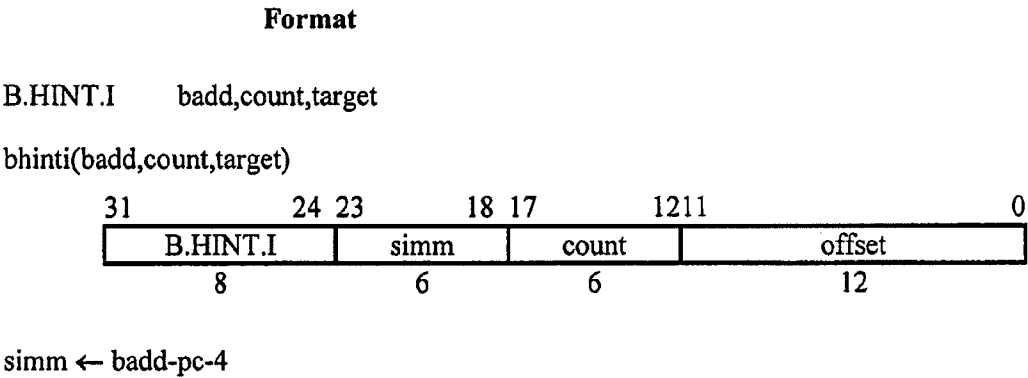


FIG. 79B

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 260 of 403**

**US 7,660,973 B2**

**Definition**

```
def BranchHintImmediate(simm,count,offset) as  
    BranchHint(ProgramCounter + 4 + (0 || simm || 02), count,  
        ProgramCounter + (offset44 || offset || 02))  
enddef
```

**Exceptions**

none

FIG. 79C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 261 of 403**

**US 7,660,973 B2**

**Operation codes**

B.I	Branch immediate
-----	------------------

**Redundancies**

B.I target	↔	B.E rc,rc,target
------------	---	------------------

**FIG. 80A**

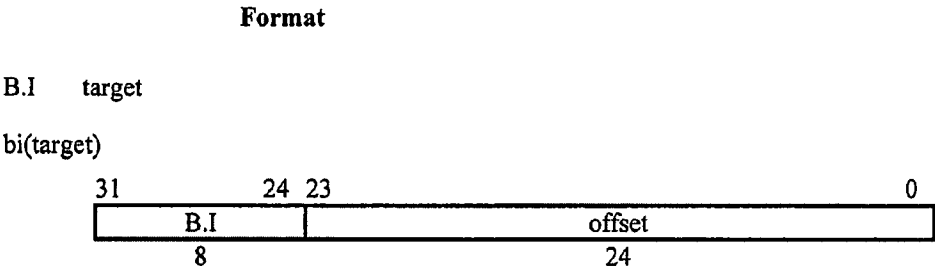


FIG. 80B

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 263 of 403**

**US 7,660,973 B2**

**Definition**

```
def BranchImmediate(offset) as
    ProgramCounter ← ProgramCounter + (offset >> 2)
    raise TakenBranch
enddef
```

**Exceptions**

none

**FIG. 80C**

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 264 of 403**

**US 7,660,973 B2**

**Operation codes**

<b>B.LINK.I</b>	<b>Branch immediate link</b>
-----------------	------------------------------

**FIG. 81A**

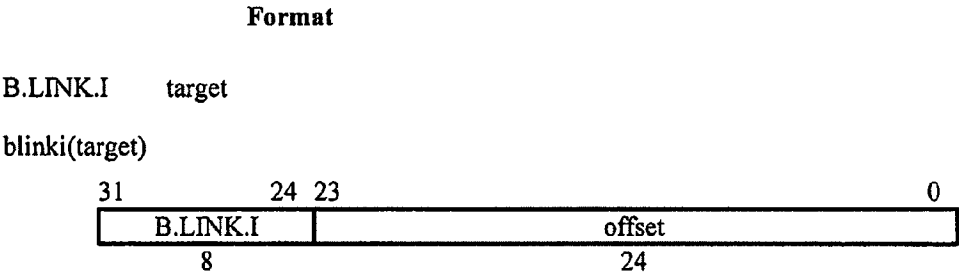


FIG. 81B



**U.S. Patent**

**Feb. 9, 2010**

**Sheet 266 of 403**

**US 7,660,973 B2**

**Definition**

```
def BranchImmediateLink(offset) as
  RegWrite(0, 64, ProgramCounter + 4)
  ProgramCounter  $\leftarrow$  ProgramCounter + (offset31:8 || offset || 02)
  raise TakenBranch
enddef
```

**Exceptions**

none

FIG. 81C

Operation codes

B.LINK	Branch link
--------	-------------

Equivalencies

B.LINK	←	B.LINK 0=0
B.LINK rc	←	B.LINK 0=rc

FIG. 82A

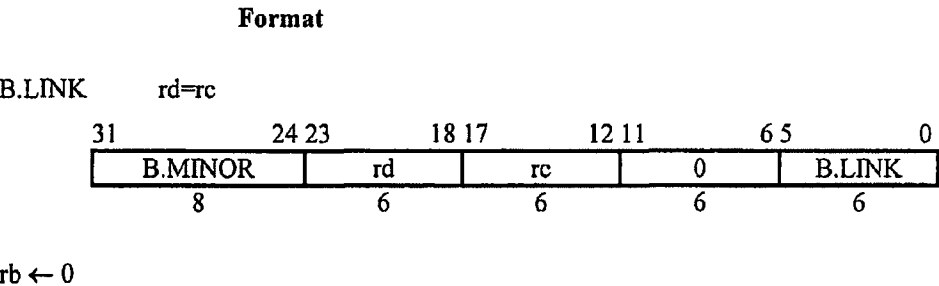


FIG. 82B

**U.S. Patent**

Feb. 9, 2010

Sheet 269 of 403

**US 7,660,973 B2****Definition**

```

def BranchLink(rd,rc,rb) as
  if rb ≠ 0 then
    raise ReservedInstruction
  endif
  c ← RegRead(rc, 64)
  if (c and 3) ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  RegWrite(rd, 64, ProgramCounter + 4)
  ProgramCounter ← c63..2 || 02
  raise TakenBranch
enddef

```

**Exceptions**

Reserved Instruction

Access disallowed by virtual address

FIG. 82C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 270 of 403**

**US 7,660,973 B2**

**Operation codes**

S.D.C.S.64.A.B	Store double compare swap octlet aligned big-endian
S.D.C.S.64.A.L	Store double compare swap octlet aligned little-endian

**FIG. 83A**

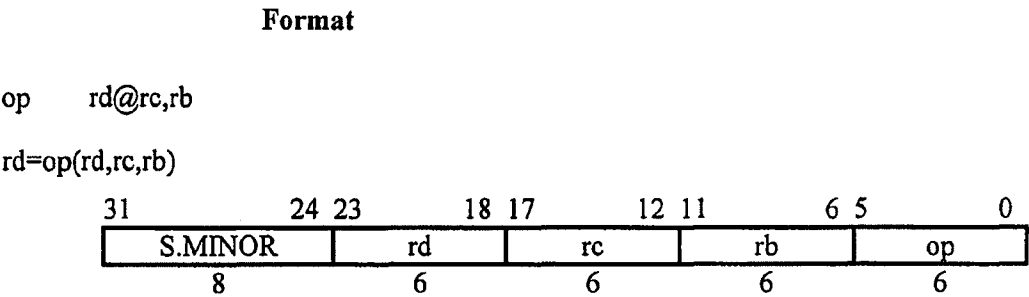


FIG. 83B

**U.S. Patent****Feb. 9, 2010****Sheet 272 of 403****US 7,660,973 B2****Definition**

```

def StoreDoubleCompareSwap(op,rd,rc,rb) as
  size ← 64
  lsize ← log(size)
  case op of
    SDCS64AL:
      order ← L
    SDCS64AB:
      order ← B
  endcase
  c ← RegRead(rc, 128)
  b ← RegRead(rb, 128)
  d ← RegRead(rd, 128)
  if (c2..0 ≠ 0) or (b2..0 ≠ 0) then
    raise AccessDisallowedByVirtualAddress
  endif
  lock
    a ← LoadMemoryW(c63..0,c63..0,64,order) || LoadMemoryW(b63..0,b63..0,64,order)
    if ((c127..64 || b127..64.) = a) then
      StoreMemory((c63..0,c63..0,64,order,d127..64)
      StoreMemory(b63..0,b63..0,64,order,d63..0)
    endif
  endlock
  RegWrite(rd, 128, a)
enddef

```

**Exceptions**

Access disallowed by virtual address  
 Access disallowed by tag  
 Access disallowed by global TB  
 Access disallowed by local TB  
 Access detail required by tag  
 Access detail required by local TB  
 Access detail required by global TB  
 Local TB miss  
 Global TB miss

FIG. 83C

**U.S. Patent****Feb. 9, 2010****Sheet 273 of 403****US 7,660,973 B2****Operation codes**

S.A.S.I.64.A.B	Store add swap immediate octlet aligned big-endian
S.A.S.I.64.A.L	Store add swap immediate octlet aligned little-endian
S.C.S.I.64.A.B	Store compare swap immediate octlet aligned big-endian
S.C.S.I.64.A.L	Store compare swap immediate octlet aligned little-endian
S.M.S.I.64.A.B	Store multiplex swap immediate octlet aligned big-endian
S.M.S.I.64.A.L	Store multiplex swap immediate octlet aligned little-endian

FIG. 84A



**U.S. Patent****Feb. 9, 2010****Sheet 274 of 403****US 7,660,973 B2****Selection**

number format	op	size	alignment	ordering
add-swap	AS	64	A	L B
compare-swap	CS	64	A	L B
multiplex-swap	MS	64	A	L B

**Format**

S.op.I.64.align.order rd@rc,offset

rd=sopi64alignorder(rd,rc,offset)

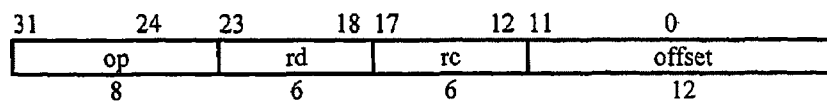


FIG. 84B

**U.S. Patent**

Feb. 9, 2010

Sheet 275 of 403

**US 7,660,973 B2****Definition**

```

def StoreImmediateInplace(op,rd,rc,offset) as
  sizE ← 64
  lsize ← log(size)
  case op of
    SASI64AL, SCSI64AL, SMSI64AL:
      order ← L
    SASI64AB, SCSI64AB, SMSI64AB:
      order ← B
  endcase
  c ← RegRead(rc, 64)
  VirtAddr ← c + (offset{5-lsize} || offset || 0{lsize-3})
  if (c{lsize-4..0} ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  d ← RegRead(rd, 128)
  case op of
    SASI64AB, SASI64AL:
      lock
      a ← LoadMemoryW(c,VirtAddr,size,order)
      StoreMemory(c,VirtAddr,size,order,d63..0+a)
      endlock
    SCSI64AB, SCSI64AL:
      lock
      a ← LoadMemoryW(c,VirtAddr,size,order)
      if (a = d63..0) then
        StoreMemory(c,VirtAddr,size,order,d127..64)
      endif
      endlock
    SMSI64AB, SMSI64AL:
      lock
      a ← LoadMemoryW(c,VirtAddr,size,order)
      m ← (d127..64 & d63..0) | (a & ~d63..0)
      StoreMemory(c,VirtAddr,size,order,m)
      endlock
  endcase
  RegWrite(rd, 64, a)
enddef

```

FIG. 84C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 276 of 403**

**US 7,660,973 B2**

**Exceptions**

Access disallowed by virtual address

Access disallowed by tag

Access disallowed by global TB

Access disallowed by local TB

Access detail required by tag

Access detail required by local TB

Access detail required by global TB

Local TB miss

Global TB miss

**FIG. 84C *continued***

**U.S. Patent****Feb. 9, 2010****Sheet 277 of 403****US 7,660,973 B2****Operation codes**

S.A.S.64.A.B	Store add swap octlet aligned big-endian
S.A.S.64.A.L	Store add swap octlet aligned little-endian
S.C.S.64.A.B	Store compare swap octlet aligned big-endian
S.C.S.64.A.L	Store compare swap octlet aligned little-endian
S.M.S.64.A.B	Store multiplex swap octlet aligned big-endian
S.M.S.64.A.L	Store multiplex swap octlet aligned little-endian

**FIG. 85A**

Selection

number format	op	size	alignment	ordering
add-swap	A.S	64	A	L B
compare-swap	C.S	64	A	L B
multiplex-swap	M.S	64	A	L B

Format

op rd@rc,rb

rd=op(rd,rc,rb)

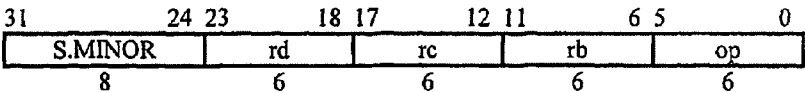


FIG. 85B

**U.S. Patent**

Feb. 9, 2010

Sheet 279 of 403

**US 7,660,973 B2****Definition**

```

def StoreInplace(op,rd,rc,rb) as
  size ← 64
  lsize ← log(size)
  case op of
    SAS64AL, SCS64AL, SMS64AL:
      order ← L
    SAS64AB, SCS64AB, SMS64AB:
      order ← B
  endcase
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 64)
  VirtAddr ← c + (b66-lsize..0 || 0lsize-3)
  if (c|size-4..0 ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  d ← RegRead(rd, 128)
  case op of
    SAS64AB, SAS64AL:
      lock
      a ← LoadMemoryW(c,VirtAddr,size,order)
      StoreMemory(c,VirtAddr,size,order,d63..0+a)
      endlock
    SCS64AB, SCS64AL:
      lock
      a ← LoadMemoryW(c,VirtAddr,size,order)
      if (a = d63..0) then
        StoreMemory(c,VirtAddr,size,order,d127..64)
      endif
      endlock
    SMS64AB, SMS64AL:
      lock
      a ← LoadMemoryW(c,VirtAddr,size,order)
      m ← (d127..64 & d63..0) | (a & ~d63..0)
      StoreMemory(c,VirtAddr,size,order,m)
      endlock
  endcase
  RegWrite(rd, 64, a)
enddef

```

FIG. 85C

**U.S. Patent**

**Feb. 9, 2010**

**Sheet 280 of 403**

**US 7,660,973 B2**

**Exceptions**

Access disallowed by virtual address

Access disallowed by tag

Access disallowed by global TB

Access disallowed by local TB

Access detail required by tag

Access detail required by local TB

Access detail required by global TB

Local TB miss

Global TB miss

**FIG. 85C *continued***